Random Forest Classifiers

IMPRO-3

Pierre Laube, Simon Scharzmeier, Christoph Viebig, Hussam Hebbo

Goals/Features

- Competitive accuracy
- Scalable / Fast
- Generalizes
- Easy evaluation
- Few hyper parameters

Algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

[1]

Algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

[1]

Algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

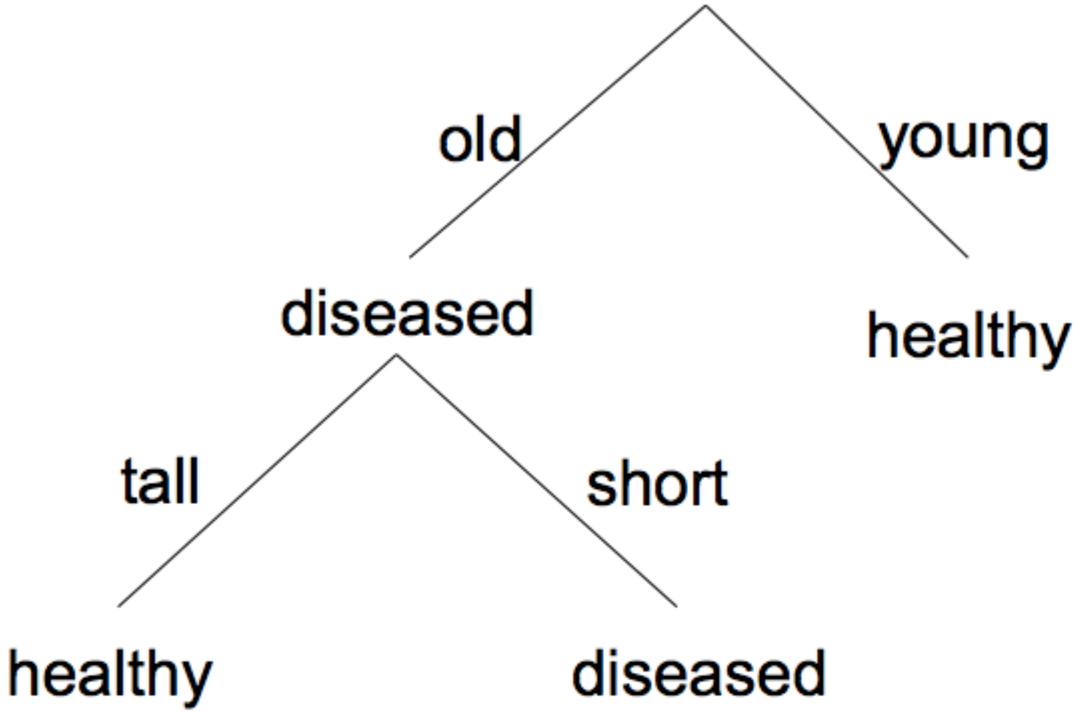
To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

[1]

Decision Tree (typical Example)



[4]

Algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

[1]

Algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

[1]

Algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

[1]

Algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

[1]

Algorithm

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:

   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.

   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.

      i. Select m variables at random from the p variables.

      ii. Pick the best variable/split-point among the m.

      iii. Split the node into two daughter nodes.

2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

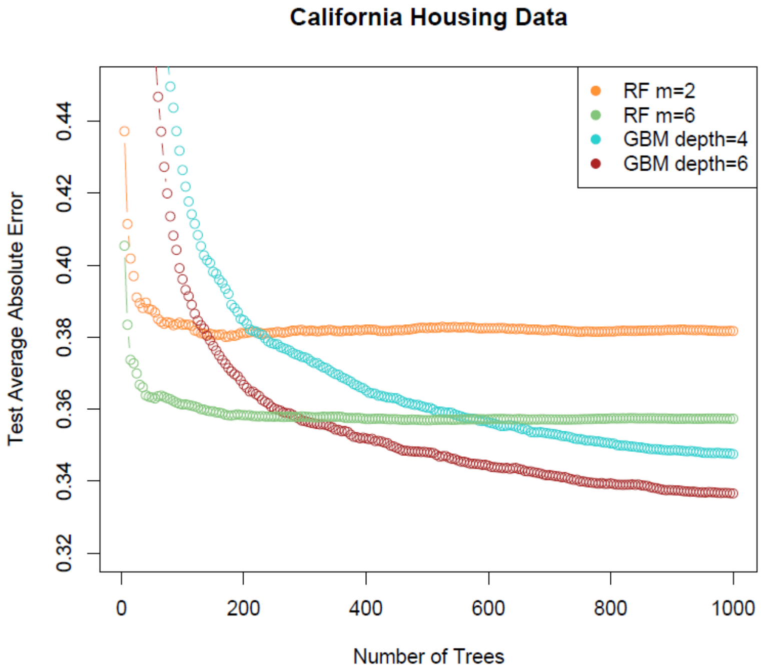
Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^B T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote}\ \{\hat{C}_b(x)\}_1^B$.

[1]

Hyper parameters

- m: Quantity of variables selected
  - Between ½√p and 2√p
- B: Quantity of trees
  - Limited gain
- N: Bootstrap Samples
  - ⅔ of the data set



[1]

Growing a random tree

0) Create first node

1) Sample features

2) Assign to node 0

3) Iterate

   a) Create child nodes

   b) Find splits

   c) Assign tuples

   d) Clean

Tuples: (data, label)

((0.1, 0.2, ...), A)
((0.1, 0.5, ...), A)
((0.3, 0.7, ...), B)
((0.3, 0.9, ...), C)

Nodes: (node, feature, split, labels)

0) Create first node

Tuples: (data, label)

((0.1, 0.2, ...), A)
((0.1, 0.5, ...), A)
((0.3, 0.7, ...), B)
((0.3, 0.9, ...), C)

Nodes: (node, feature, split, labels)

(1, , , [ ])

1) Sample features

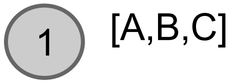
Tuples: (data, label)

((0.1, 0.2), A)
((0.1, 0.5), A)
((0.3, 0.7), B)
((0.3, 0.9), C)

Nodes: (node, feature, split, labels)

(1, , , [ ])

2) Assign to node 0



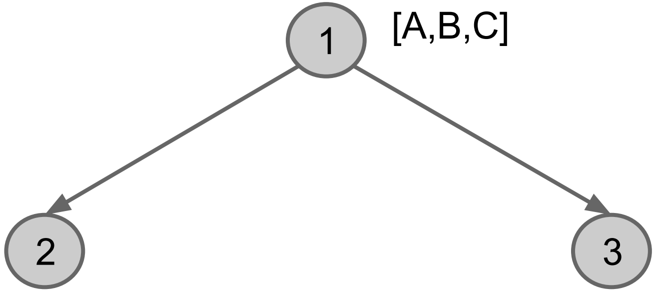
Tuples: (node, data, label)

(1, (0.1, 0.2), A)
(1, (0.1, 0.5), A)
(1, (0.3, 0.7), B)
(1, (0.3, 0.9), C)

Nodes: (node, feature, split, labels)

(1, , , [A, B, C])

a) Create child nodes



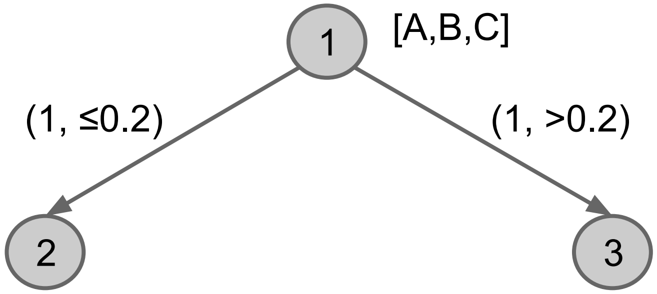
Tuples: (node, data, label)

(1, (0.1, 0.2), A)
(1, (0.1, 0.5), A)
(1, (0.3, 0.7), B)
(1, (0.3, 0.9), C)

Nodes: (node, feature, split, labels)

(1, , , [A, B, C])
(2, , , [ ])
(3, , , [ ])

b) Find splits



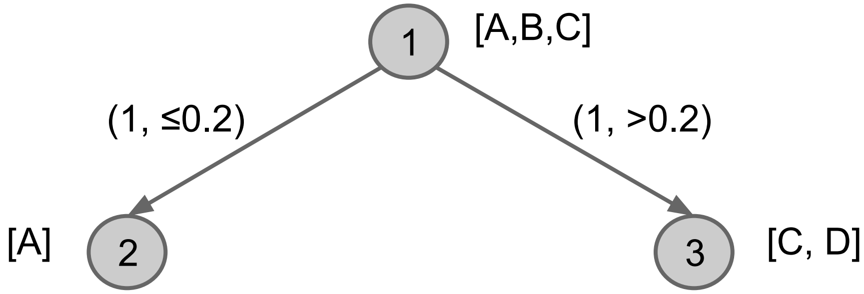
Tuples: (node, data, label)

(1, (0.1, 0.2), A)
(1, (0.1, 0.5), A)
(1, (0.3, 0.7), B)
(1, (0.3, 0.9), C)

Nodes: (node, feature, split, labels)

(1, 1, 0.2, [A, B, C])
(2, , , [ ])
(3, , , [ ])

c) Assign tuples



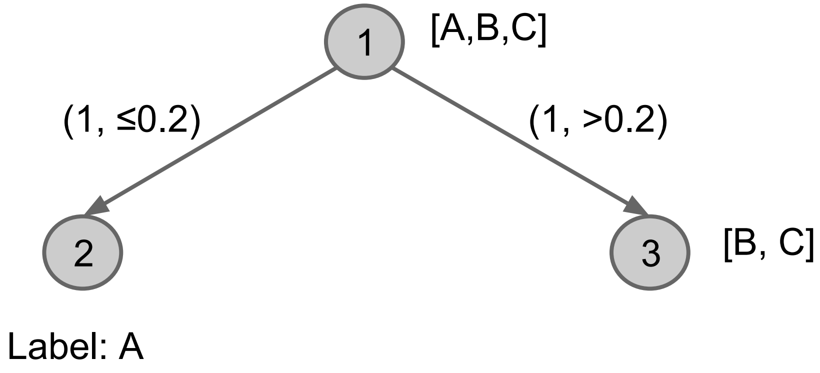
Tuples: (node, data, label)

(2, (0.1, 0.2), A)
(2, (0.1, 0.5), A)
(3, (0.3, 0.7), B)
(3, (0.3, 0.9), C)

Nodes: (node, feature, split, labels)

(1, 1, 0.2, [A, B, C])
(2, , , [A])
(3, , , [B, C])

d) Clean



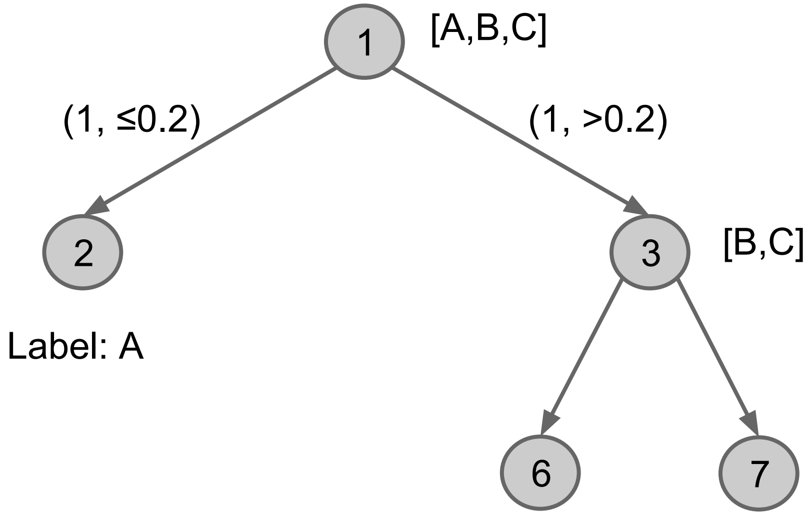
Tuples: (node, data, label)

(3, (0.3, 0.7), B)
(3, (0.3, 0.9), C)

Nodes: (node, feature, split, labels)

(1, 1, 0.2, [A, B, C])
(2, , , [A])
(3, , , [B, C])

a) Create child nodes



Tuples: (node, data, label)

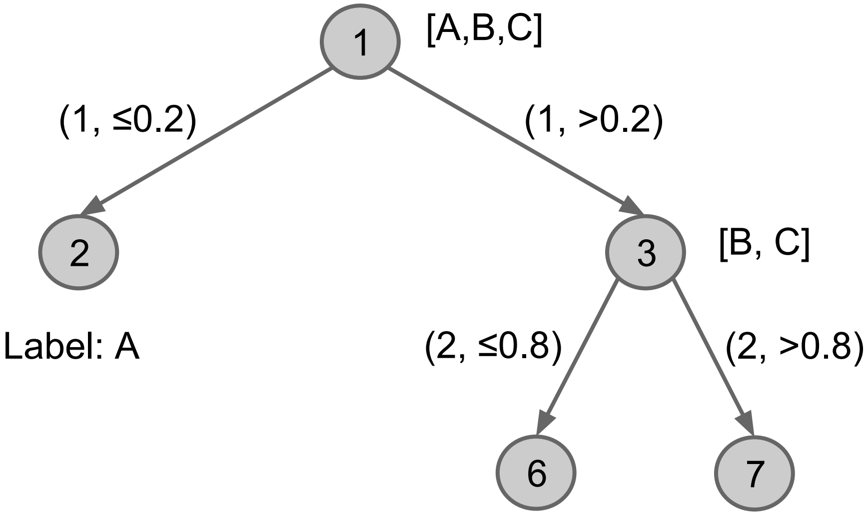
(3, (0.3, 0.7), B)
(3, (0.3, 0.9), C)

Nodes: (node, feature, split, labels)

(1, 1, 0.2, [A, B, C])
(2, , , [A])
(3, , , [B, C])

(6, , , [ ])
(7, , , [ ])

b) Find splits



Tuples: (node, data, label)

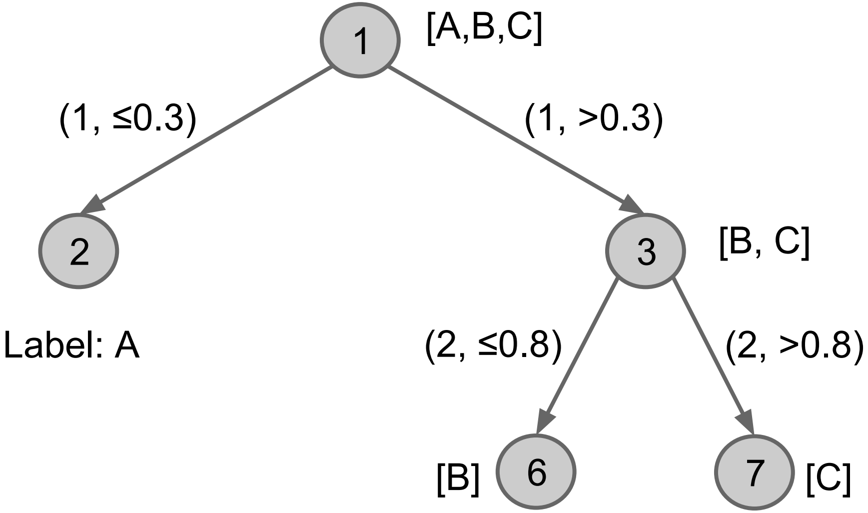
(3, (0.3, 0.7), B)
(3, (0.3, 0.9), C)

Nodes: (node, feature, split, labels)

(1, 1, 0.2, [A, B, C])
(2, , , [A])
(3, 2, 0.8, [B, C])

(6, , , [ ])
(7, , , [ ])

c) Assign tuples



Tuples: (node, data, label)

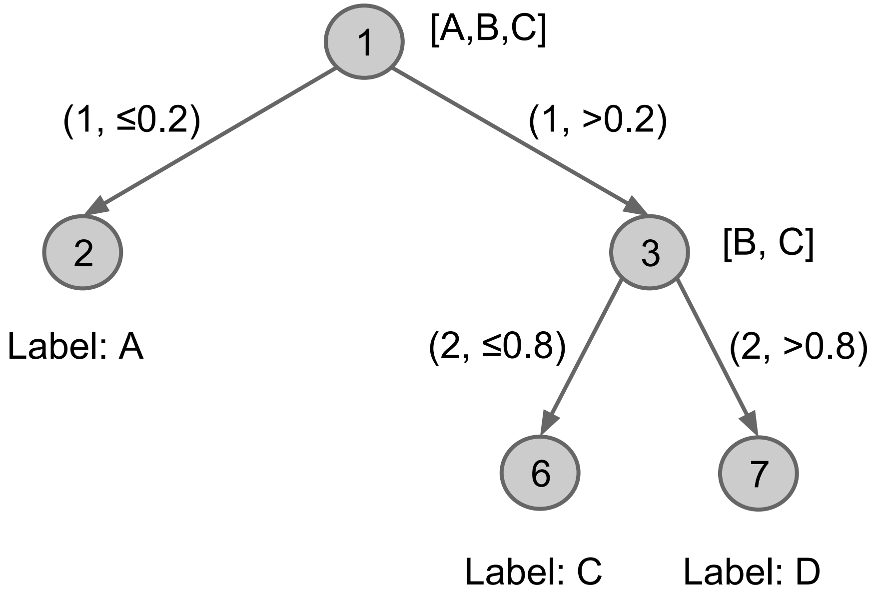
(6, (0.3, 0.7), B)
(7, (0.3, 0.9), C)

Nodes: (node, feature, split, labels)

(1, 1, 0.2, [A, B, C])
(2, , , [A])
(3, 2, 0.8, [B, C])

(6, , , [B])
(7, , , [C])

d) Clean



Tuples: (node, data, label)

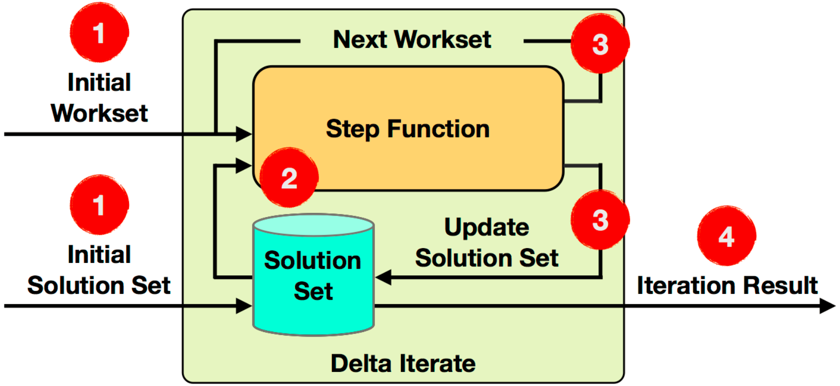
Nodes: (node, feature, split, labels)

(1, 1, 0.2, [A, B, C])
(2, , , [A])
(3, 2, 0.8, [B, C])

(6, , , [B])
(7, , , [C])

Delta Iterations

0) Create first node
1) Sample features
2) Assign to node 0
3) Iterate
   a) Create child nodes
   b) Find splits
   c) Assign tuples
   d) Clean



Work Set:	Data and node assignment [(node, data, label)]
Solution Set:	Nodes [(id, [label, quantity], label)]

Delta Iterations

0) Create first node → 1

1) Sample features ⎫
2) Assign to node 0 ⎭ 1

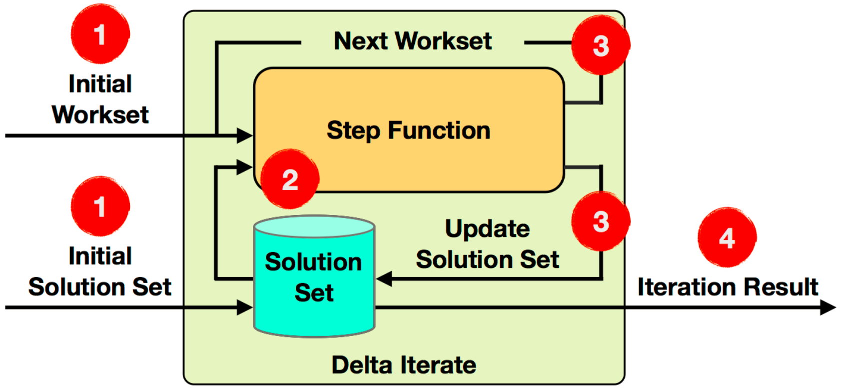
3) Iterate

   a) Create child nodes

   b) Find splits

   c) Assign tuples

   d) Clean



Work Set: Data and node assignment
[(node, data, label)]

Solution Set: Nodes
[(id, [label, quantity], label)]

Delta Iterations

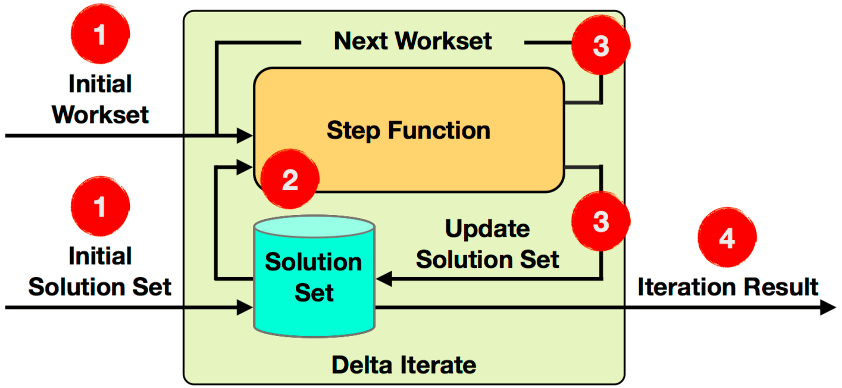
0) Create first node → 1

1) Sample features
2) Assign to node 0
} 1

3) Iterate → Step function

  a) Create child nodes
  b) Find splits
  } 3

  c) Assign tuples
  d) Clean
  } 3



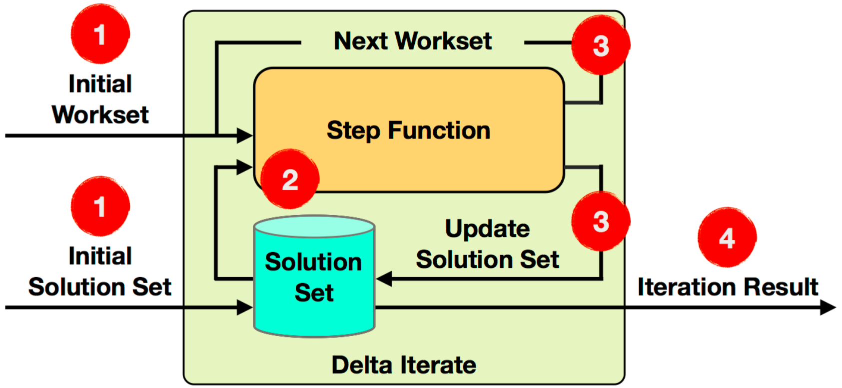
Work Set: Data and node assignment
[(node, data, label)]

Solution Set: Nodes
[(id, [label, quantity], label)]

Delta Iterations

0) Create first node → 1

1) Sample features
2) Assign to node 0 } 1
3) Iterate → Step function
    a) Create child nodes
    b) Find splits } 3
    c) Assign tuples
    d) Clean } 3
4) If empty work set → 4



Work Set: Data and node assignment
[(node, data, label)]

Solution Set: Nodes
[(id, [label, quantity], label)]

How to actually find Splits?

Using Gini Index selection (splitting) measure

The Gini index considers a binary split for each attribute

It forces any tree to be binary

What are possible Splits?

Assume <, =, > are applicable on each attribute/feature

→ **K-1** possible splits *(K distinct values in feature set)*

$$[v_0, v_1, v_2, \ldots, v_K]$$

$$v_i < v_{i+1}$$

What are possible Splits?

For discrete categorizable data:

We will assume non-categorical features!

$$2^N - 2$$

2^N: all possible combinations

2: the full and the empty sets

What is a good Split?

- 3 common measures
- All based on heuristic arguments
  - **Missclassification** *(Breiman)*
  - ***Information Gain*** *(ID.3/C4.5)*
  - ***Gini Impurity*** *(CART)*

Best choice of method is not generalizable [5] - we use Gain for scala- and Gini for stratosphere- version.

Common for all of them

- For each possible 2-split:
  - Calc class-distributions
  - Derive heuristic measure
- Choose best by measure

Common for all of them

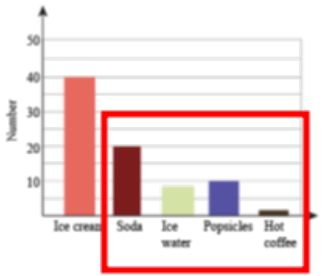
Example:

Missclassification:

$$[(\vec{x}, y) \in D]$$

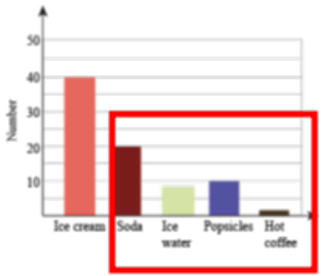
$$[(\vec{x}, y) \in D | x_a \leq splitval] \qquad [(\vec{x}, y) \in D | x_a > splitval]$$

Left Class Distribution



Missclassification

Right Class Distribution



Missclassification

Gini Index

The gini for the whole data

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

Gini for each possible split

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2).$$

Gini for each attribute

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

Example

RID	*age*	*income*	*student*	*credit_rating*	*Class: buys_computer*
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

Example

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

Possible splits:

{low, medium, high},

{low, medium}, {low, high}, {medium, high},

{low}, {medium}, {high},

{}

Example

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

Possible splits:

{low, medium, high},
{low, medium}, {low, high}, {medium, high},
{low}, {medium}, {high},
{}

Example

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

Possible splits:

{low, medium}, {low, high}, {medium, high},
{low}, {medium}, {high},

Example

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

Possible splits:

{low, medium}, {low, high}, {medium, high},

{low}, {medium}, {high},

$$Gini_A(D) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2).$$

$$\begin{aligned}
&Gini_{income \in \{low,medium\}}(D) \\
&= \frac{10}{14}Gini(D_1) + \frac{4}{14}Gini(D_2) \\
&= \frac{10}{14}\left(1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2\right) + \frac{4}{14}\left(1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2\right) \\
&= 0.450 \\
&= Gini_{income \in \{high\}}(D).
\end{aligned}$$

Example

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

Possible splits:

{low, medium}, {low, high}, {medium, high},
{low}, {medium}, {high},

$$Gini_A(D) = \frac{|D_1|}{|D|} Gini(D_1) + \frac{|D_2|}{|D|} Gini(D_2).$$

Gini index for the possible splits:

{low, medium} or {high} = 0.450
{low, high} or {medium} = 0.315
{medium, high} or {low} = 0.300

Example

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

Possible splits:

{low, medium}, {low, high}, {medium, high},
{low}, {medium}, {high},

$$Gini_A(D) = \frac{|D_1|}{|D|}Gini(D_1) + \frac{|D_2|}{|D|}Gini(D_2).$$

Gini index for the possible splits:

{low, medium} or {high} = 0.450
{low, high} or {medium} = 0.315
{medium, high} or {low} = 0.300

Example

RID	age	income	student	credit_rating	Class: buys_computer
1	youth	high	no	fair	no
2	youth	high	no	excellent	no
3	middle_aged	high	no	fair	yes
4	senior	medium	no	fair	yes
5	senior	low	yes	fair	yes
6	senior	low	yes	excellent	no
7	middle_aged	low	yes	excellent	yes
8	youth	medium	no	fair	no
9	youth	low	yes	fair	yes
10	senior	medium	yes	fair	yes
11	youth	medium	yes	excellent	yes
12	middle_aged	medium	no	excellent	yes
13	middle_aged	high	yes	fair	yes
14	senior	medium	no	excellent	no

Overall Gini index for all attributes:

- {age} with {youth, senior} = 0.375
- {income} with {medium, high} = 0.300
- {student} = 0.367
- {credit_rating} = 0.429

$$Gini(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459.$$

$$\Delta Gini(A) = Gini(D) - Gini_A(D).$$

Gini impurity (splitting criterion):

- {age} with {youth, senior} = 0.084
- {income} with {medium, high} = 0.159
- {student} = 0.092
- {credit_rating} = 0.03

Splits In Stratosphere

For each Delta-Iteration (Tree-Layer):

- calc local histograms for each node ->
- combine local histograms ->
- group by (node, dim) ->
- based on histograms, locally calc Gini-Idx for every possible Split and find best split for (node, dim)
- aggregate best dim-splits to find best splits of each (node)
- yield the best Splits

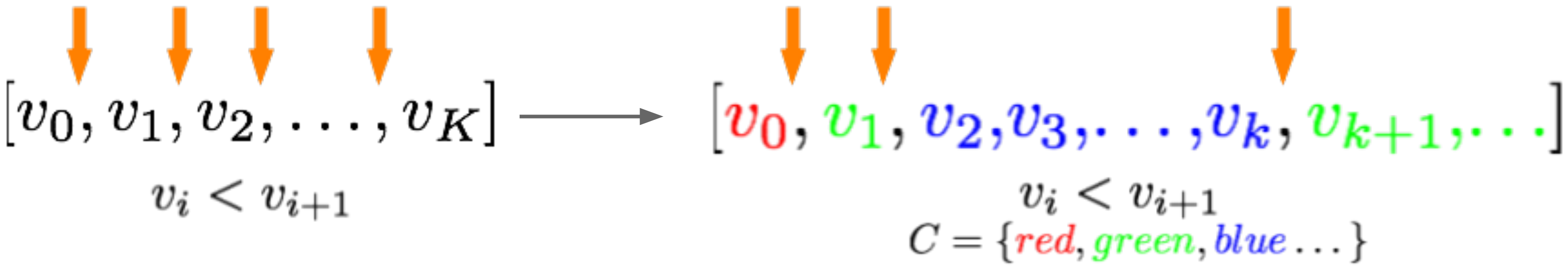
Problem?

Yes! Too many distinct values to hold histograms in memory

Solution: somehow reduce the # of distinct values / number of possible Splits

Optimization for Gini-Index (1)

try to reduce number of possible splits with **global pre-sorting**



Optimization for Gini-Index (2)

based on MLib implementation of DecisionTrees [6,7]:

- infer possibly adequate quantization-intervals using downsampling
- use the intervals to bin values

→ We will give this variant a shot

References

[1] Hastie, T.; Tibshirani, R.; Friedman, J.: The Elements of Statistical Learning. New York, NY, USA: Springer New York Inc., 2001.

[2] Hastie, T.; Tibshirani, R.; Witten, D; and James, G.: An Introduction to Statistical Learning. New York, NY, USA: Springer New York Inc., 2013.

[3] Breiman, L.: Random forests. Machine learning 45 , no. 1 (2001): 5-32.

[4] ETH Zurich: Random forests. Applied Multivariate Statistics – Spring 2012.

[5] Raileanu; Elena L.; Stoffel K.:Theoretical comparison between the gini index and information gain criteria. Annals of Mathematics and Artificial Intelligence 41.1 (2004): 77-93.

[6] http://spark.apache.org/docs/latest/mllib-decision-tree.html

[7] https://github.com/apache/spark/tree/master/mllib/src/main/scala/org/apache/spark/mllib/tree